Optimal Execution of Orders in a Limit Order Book

Morgan Kidd, Yashoraj Tyagi, Maximilien Lamberti,

Gary Finkelstein, Xian Li

March 2020

Abstract

This paper explores the optimal order execution problem using reinforcement learning. We implement an empirically driven limit order book simulator based on Vyetrenko (2019) and Spooner (2018) as the learning environment in which the agent is allowed to place limit as well as market orders. By varying the parameters of our reward function, which accounts for implementation shortfall, inventory risk and order spamming, we successfully train four distinct trading agents. Characteristic distributions of implementation shortfall are evaluated and agent trading behaviour is explored in detail. The classic Almgren & Chriss framework (2000) as well as the execution strategy presented by Guo, Larrard, and Ruan (2017), which we refer to as the *Guo Model*, are implemented as baseline comparisons. We find that the Guo model outperforms our reinforcement learning approach, which in turn outperforms the Almgren & Chriss model.

Acknowledgements

We thank Professor Hendershott for supervising our AFP project. Additionally, we thank Professor Hendershott and Professor Livdan for teaching us about market microstructure and order execution. We are grateful to Dr. Reiner who steered us clear from promising but intangible topics. We also say thank you to our fellow students Robert Sharow and Olivier Fortineau who helped us access an uncooperative API. Finally, we thank the MFE Office and particularly Aaron Nebres for making the data infrastructure available and expanding disk space when the going got tough.

Contents

1 Introduction

In the finance industry, one of the problems participants face is the optimal order placement and execution problem. Oftentimes, traders will need to liquidate or take positions in an asset where the price at which the trade is executed at is critical for the profitability of the strategy. A trader can opt to trade aggressively, placing market orders so as to reach a target position quickly. However, in cases where the market suffers from liquidity constraints, an aggressive execution strategy could eat into a substantial amount of the limit order book, resulting in execution at unfavourable prices. Furthermore, large market orders can cause an imbalance in the demand/supply dynamics of the limit order book resulting in prices moving against the trader. Alternatively a trader can trade less aggressively, placing limit orders in the order book in an attempt to ensure execution at certain prices. However, in this case, the trader faces high execution risk where the limit orders may take too long to be executed, an issue if the position needs to be entered into quickly, or the possibility of not being executed at all, resulting in the trader missing out on any returns they were expecting from their target portfolio.

There have been several papers on optimal order execution over the past two decades. Bertsimas and Lo (1998) published their paper on optimal order execution, assuming a static price impact function in order to develop an optimal execution strategy (Bertsimas and Lo, 1998)[1]. Almgren and Chriss (2000), having proposed an arithmetic random walk price process for the underlying asset, discretized the order execution problem and modelled trades as having both a temporary and permanent linear impact on price (Almgren and Chriss, 2000)[2]. Obizhaeva and Wang (2005) expanded on this impact function by introducing a resilience term, and proposing a non-static price impact function using supply and demand dynamics (Obizhaeva and Wang, 2005)[3]. Benazzoli and Persio (2014) built on Almgren and Chriss' optimal execution model by modeling the drift and volatility of the underlying asset as a Markov process (Benazzoli and Persio, 2014)[4]. Guo, Larrard, and Ruan (2017) has addressed the optimal placement problem through work on optimizing the probability of orders being filled (Guo, Larrard, and Ruan, 2013)[5].

This paper uses reinforcement learning as an approach to solve the optimal order placement and execution problem. Due to the complexities around modeling limit order book dynamics, specifically, market participant behavior, drift, and asset liquidity, historical order book and trade data is used to model the limit order book. A reinforcement learning agent is trained by mapping order book conditions to its corresponding optimal trade action. Compared to traditional approaches to the optimal execution problem, this paper attempts to develop a more robust and optimal order execution model through the use of historical limit order book data and reinforcement learning techniques whilst limiting the need for assumptions on order book dynamics.

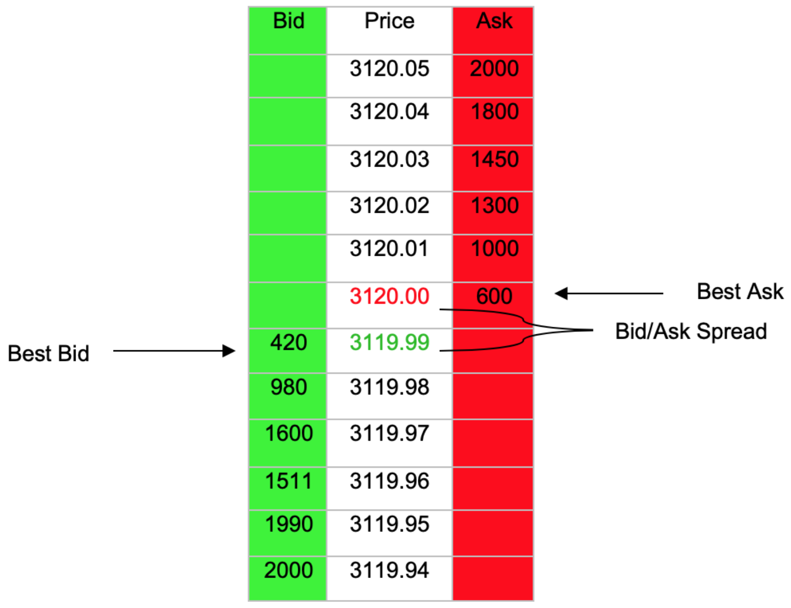
The paper is organized as follows: Section 2 explains key concepts and features of the limit order book, the different types of orders that can be executed in the market and how the reinforcement learning agent fits into the problem. Section 3 discusses the data and methodology used to simulate market behavior, in particular, how historical limit order book data is used to model trade impact on asset prices. Section 4 and 5 details traditional approaches to the optimal execution problem, specifically Almgren and Chriss, as well as Guo's more recent advances in the topic. Section 6 focuses on the reinforcement learning approach, its implementation, and results. Sections 7 and 8 present a discussion and conclusion to the paper's research.

2 Explanation of Key Concepts

I The Limit Order Book

A limit order book matches buyers and sellers of a financial instrument by keeping record of the buy and sell orders placed by market participants. Buyers of the financial instrument place buy orders in a specified quantity at a specified price, known as the bid price, whilst sellers place sell orders in a specified quantity with price referred to as the ask price. The difference between the best bid and best ask price is referred to as the bid-ask spread (Hollified, Miller, and Sandas, 2004)[6]. For a given limit order book, there is a specified lot size and tick size. The lot size specifies the minimum amount of a specified financial instrument that can be traded whilst the tick size specifies the minimum allowed price increment for which a buy or sell order can be placed.

Figure 1 illustrates a limit order book of an asset with a tick size of $0.01.



Bid	Price	Ask
	3120.05	2000
	3120.04	1800
	3120.03	1450
	3120.02	1300
	3120.01	1000
	3120.00	600
420	3119.99	
980	3119.98	
1600	3119.97	
1511	3119.96	
1990	3119.95	
2000	3119.94	

Figure 1: Limit Order Book of an Asset

Figure 2 below illustrates a limit order books volume dynamics:

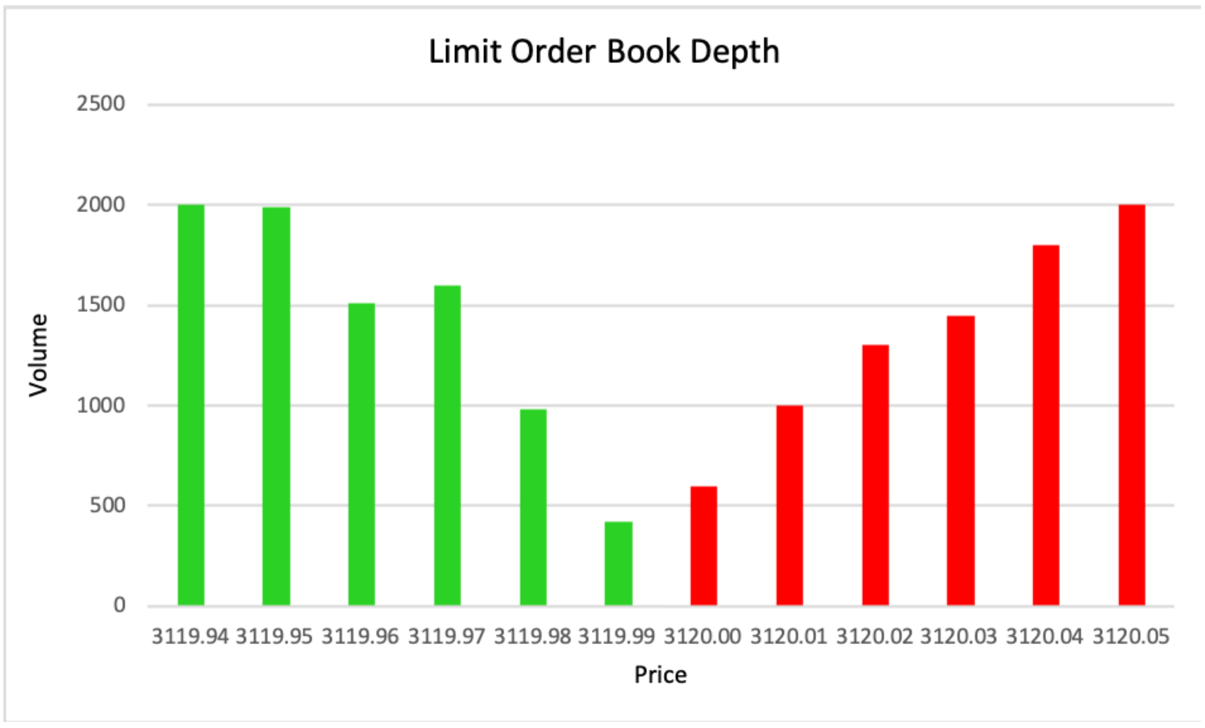


Figure 2: Limit Order Book Volume Dynamics

II Limit and Market Orders

Limit orders and market orders refer to the way in which a trade is to be executed in a limit order book.

A market buy or sell order is an instruction to immediately execute on the buying or selling of a specified quantity of an asset (Hollified, Miller, and Sandas, 2004)[6]. Should the lot size of a market buy order be larger than the available quantity offered at the best asking price, the buy order is filled at the best ask in the quantity offered and then filled at the next best offers in its remaining quantity. Market sell orders are executed in the same manner against the best bids. In this way, market orders are prone to slippage, where the price at which a trade is executed differs to the current market price as the trade eats into the order book depth.

A limit order is an instruction to buy or sell a specified quantity of an asset at a specified price (Hollified, Miller, and Sandas, 2004)[6]. Execution of the instruction is not immediate, rather buy limit orders are executed when the best ask price is at or below the buy limit order price, and sell limit orders are executed when the best bid price is at or above the sell limit order price (Hollified, Miller, and Sandas, 2004)[6]. Limit orders are prone to execution risk, in the sense that a limit order may not be executed at all should the best bid or best ask not match the buy or sell limit order price (Hollified, Miller and Sandas, 2004)[6].

Figure 3 illustrates a sell market order being executed in the limit order book and eating through the bids placed.

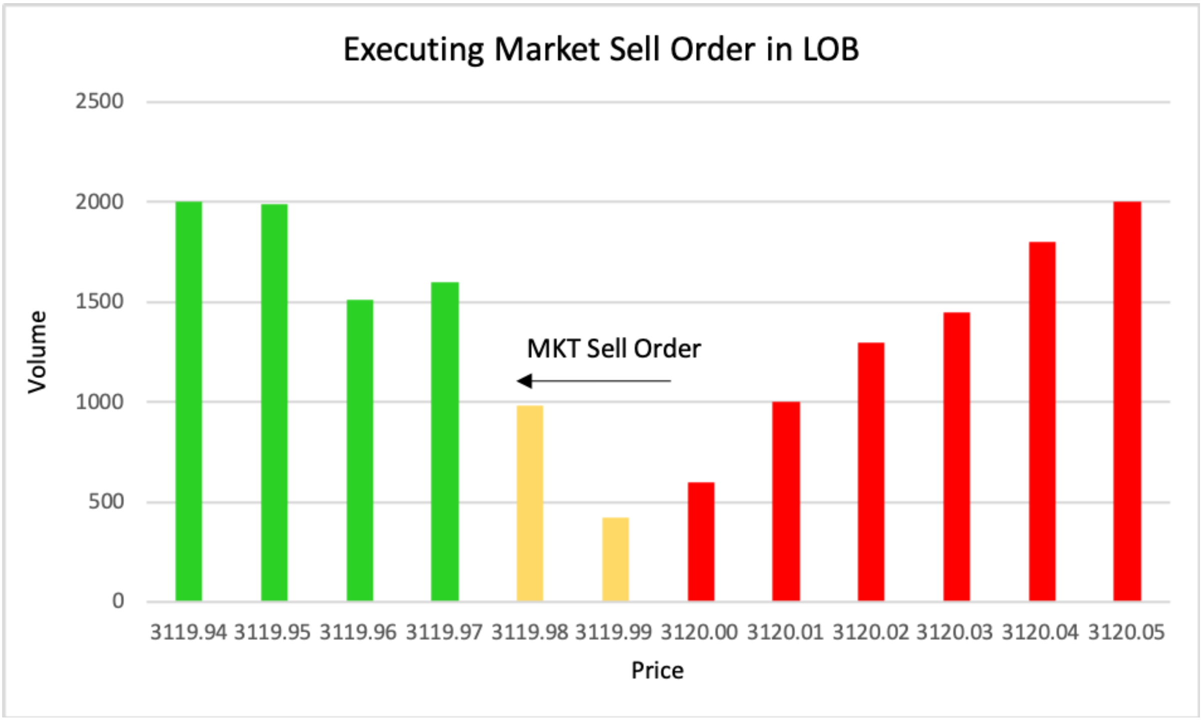


Figure 3: Market order executed against the limit order book

In essence, successful limit orders are profitable, since the exchange occurs at a favourable price relative to the best bid and ask at the time of placing the order. Additionally, depending on the depth of the limit order, the limit order may earn a rebate, in the sense that a the order book depth difference is earned (Guo, Larrard, and Ruan, 2017)[5]. Market orders, on the other hand, are penalized by the bid/ask spread. However, market orders have an execution guarantee, so the target trade amount can almost always be completed within the allotted time.

III Order Execution, Order Placement and the Agent

Optimal order execution refers to the best method to split large orders into smaller ones in order to minimize shortfall (Guo, Larrard, and Ruan, 2017)[5]. Optimal order placement refers to placing the best type of order given the size of a trade to execute, either in the form of a limit order or a market order (Guo, Larrard, and Ruan, 2017)[5]. The two are closely tied together in that for the most optimal trade, a trader needs to know when to switch between limit orders and market orders, as well as knowing how to split their market orders in a way that minimizes shortfall given the amount of time left to execute their trades.

A reinforcement learning agent is an entity that makes decisions based on its observations in its environment. In this case, the environment is the limit order book. The aim of this paper is to train the agent to be able to optimally place limit and market orders to liquidate its assets, given a set amount of time within which to execute, the state of the limit order book at each moment in time, the amount of inventory the agent has left to trade and the price movement of the underlying asset.

IV Market Order Impacts

In order to visualize the price impacts of trades, we look at the return distribution following a buy or sell order of size 100. The reason we chose order sizes of 100 is because the majority of historical orders are of size 100, accounting for 53% of all executed market orders in the observation period.

Figure 4 shows the change in prices from averaging 20,000 historical size 100 trades.

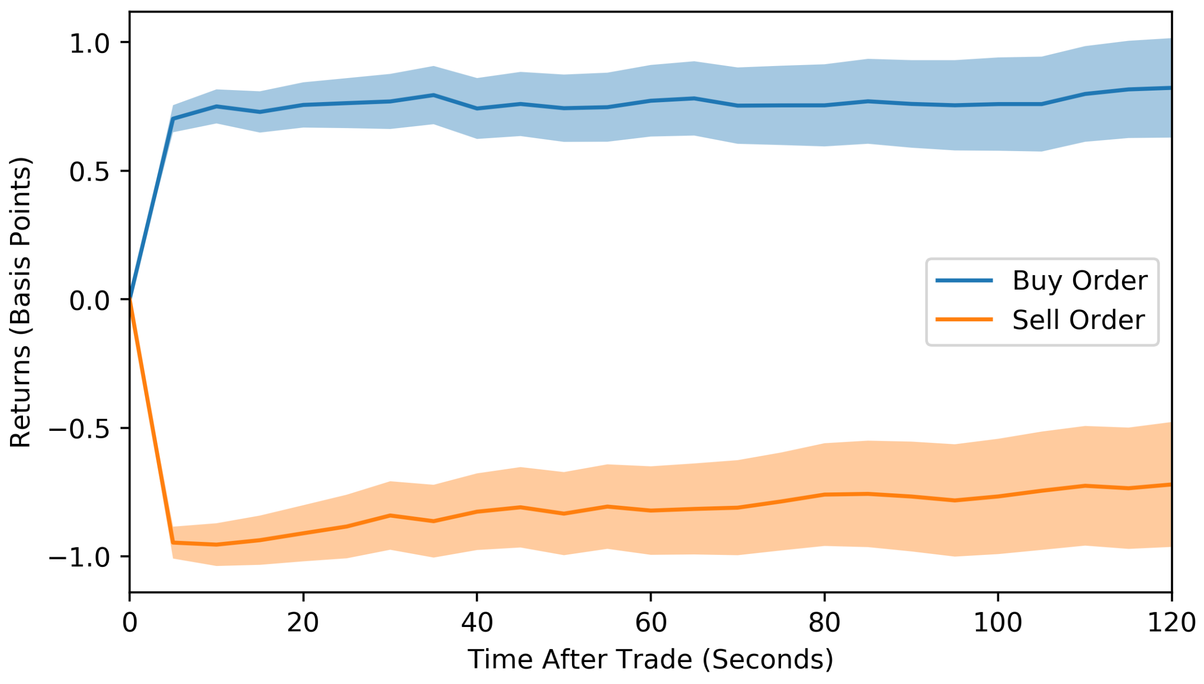


Figure 4: Average 120 Second Price Impact of Executing an Order of Size 100

Characteristic for return data, we see from the plot that the variance of returns increases with time. The slight upward trend of the two curves are fragments of the long run price drift of the underlying asset. From the slope of the two curves, we can infer some mean reverting behavior as the two lines angle towards each other.

3 Methodology

I Data

This paper uses OneTick's historic market and trading data for Apple (AAPL) stock. The Apple order book and trade data was selected because it is one of the most frequently traded stocks on the NASDAQ. Observing frequent trades in the market is a desirable property for the limit order book simulator used in this paper.

Our order book data consists of five second snapshots of NASDAQ ITCH data for Apple consisting of the aggregate volume for the first ten tick levels on the ask and the bid side. We also have timestamped realized trades that happened over the observation period. Each trade data point consists of the volume traded and the direction of the trade, either a buy or a sell.

The data ranges from January 2012 to December 2015. To create a windowed data set, we bootstrap-sample trading windows randomly between January 2012 and December 2014. The 2015 data is exclusively used as an out-of-sample test set.

II The Limit Order Book Simulator

For greatest accuracy, the quality of an execution strategy should be evaluated in a real life trading environment. However, due to constraints like capital and slowness of real time execution, this approach is usually limited to institutional market participants. For strategy evaluation, we therefore settle for an approach which simulates the limit order book dynamics. Compared to a real life implementation, a simulation-based approach will always suffer from simplifying assumptions about the market dynamics. However, it provides a flexible way to back-test strategies in parallel across multiple parameters, asset classes and historical periods at zero implementation cost. Simulation-based approaches are not only popular in the field of market microstructure, but also in reinforcement learning. Even for moderately complex problems, reinforcement learning agents need a large amount of data and training to converge to optimal policies. Rarely is reinforcement learning performed in real time.

There exists a large variety of structural models which aim to emulate the dynamics of the limit order book. Popular implementations for example include Hawke's process-based models (Lu and Abergel 2017)[7], the Queue Reactive Model (Huang, Lehalle, and Rosenbaum 2014)[8], agent-based models and those implemented by other AFP groups (Sharow et al. 2020). These models make assumptions on the arrival rate of orders, the functional form of a trade's price impact, or the stochastic processes driving the price series. While these models each try to capture particular stylized facts and statistics of the limit order book (Vyetrenko, Byrd et al. 2019)[9], one can argue they are insufficient at evaluating order execution strategies objectively. Due to the freedom of choosing between models and assumptions, the performance of any particular execution strategy is highly dependent on the modeler's choice. For example, a limit order book model which uses an exponentially decreasing impact function would favor solutions similar to the optimal execution strategy presented by Obizhaeva and Wang 2014[3]. We make minimal assumptions on the underlying dynamics of the limit order book and, instead, implement an approach driven chiefly by historical data. This approach was first implemented by Spooner et al. (2018), and then extended upon by Vyetrenko and Xu (2019). While guided by their approach, we make changes to better fit the simulator to our use case.

The simulation we employ is essentially a replay of historical limit order book and realized trades data. When our agent places an order, we execute the trade in line with historically realized trades, that is, pretending that trades placed by other market participants are instead placed by our agent. While this is an elegant way of capturing realistic price dynamics, price impact and execution metrics, difficulties arise from modelling queue priority of limit orders as well as dealing with periods in which there were no realized historical trades. In the following paragraphs we detail the exact implementation of the limit order book simulator and discuss the modelling choices and illuminating edge cases.

II.1 Market Orders

The simulator periodically iterates through snapshots of the historical limit order book at a frequency of ΔT. At each period t, the agent is provided with the first N levels of tick data at t, as well as an array of the execution prices and volume of the historically realized trades within the period $t - \Delta T$ and t.

Execution of market orders is handled in a straightforward way. A sell order placed at t is executed against the historically observed bid side of the limit order book at that instant. If the order size is greater than the first tick level, the remaining volume is executed in order of most desirable price against the volume in the remaining tick levels. We therefore capture the exact transaction price of a market order if it were executed at that instant in time against the true limit order book.

While the above execution generates an empirically sound value for the transaction price, the market order is only theoretical and doesn't lead to any price impact. To address this issue, we assume the existence of two different types of market orders: *small* and *large* volume. The distinction between small and large volume orders originates from Vyetrenko (2019)[9]. We define these two order types for buy and sell orders in relation to the volume at the relevant first tick level:

1. Large Market Sell Order:

$$\frac{V_{bid}}{O_{sell}} \leq C \tag{1}$$

2. Small Market Sell Order:

$$\frac{V_{bid}}{O_{sell}} > C \tag{2}$$

3. Large Market Buy Order:

$$\frac{V_{ask}}{O_{buy}} \leq C \tag{3}$$

4. Small Market Sell Order:

$$\frac{V_{ask}}{O_{buy}} > C \tag{4}$$

Where $V_{ask/bid}$ is the volume at the best ask or bid price, O is the volume of the agent's market order, and C is a constant model hyperparameter.

When placing market orders which qualify as small, we assume price impact is negligible and do not make adjustments to the price series. Hence, the next market state of the simulator is the historically observed order book at $t + \Delta T$ as well as the realized trades between t and $t + \Delta T$.

For large orders, however, instead of proceeding to the next period historical data, we jump the market state to the period after which the next large historic market order was placed. For example, say we place a large market order at period t and there are no historic large market orders between t and $t + \Delta T$. The simulator then jumps to period t', for $t' > t$, where the next large historic market order was placed between t' and $t' + \Delta T$, and resumes the market from that period onward. We therefore don't make explicit assumptions about the shape of the price impact function, as is common in optimal execution frameworks, but bootstrap off the historical price impact of similar orders. With this approach we aim to capture adverse price behaviour when placing orders at limited liquidity.

Our approach differs from Vyetrenko (2019) in that we always evaluate execution price against the historic limit order book, whereas Vyetrenko et al. replace the transaction price of large volume orders with the historic transaction price of the market order in period t' (Vyetrenko and Xu, 2019)[10].

II.2 Limit Orders

The order book simulator allows for the placement of limit orders. A critical aspect when modelling limit order placement is accounting for time priority. For example, for two orders placed at the same tick level, the order which was placed first will get executed first, i.e. *"first-in-first-out"*. When placing the agent's orders in the limit order book, we therefore need to have a model of the agent's position in the queue. To this end, we follow Spooner et al. (2018) in their order book simulation (Spooner, Fearnley, Savani, Koukorinis 2018)[11].

When placing a limit order at a tick level, the agent's order is naturally placed at the back of the order queue for that tick level. In the next period, if we observe that the queue sized increased, we add those orders behind the agent's orders. If the queue sized decreased, this means that orders have been cancelled. Since we only have aggregate information about the order volume at a tick level, we model the cancellation occurring with uniform probability throughout the queue. For example, if there is a volume of 100 in front our agent's order and 200 behind it, the probability that the cancellation occurs in front of the agent's order is 1/3 and the probability that the cancellation occurs behind the agent's order is 2/3.

A complicating matter in the addition and cancellation estimation is the case when there are market orders being executed against the queue at the same time when the queue size is changing. Based on observed transaction volume and volume depth at order book levels, we estimate the amount of volume executing against each level of the limit order book. We net the executed transaction volume at the tick level against observed gross period-to-period change in tick level volume to arrive at the final estimate of the net additions and cancellations at that tick level.

Note that based on the aggregate nature of the data, the simulator can only perform either additions or cancellations to the queue, but not both. Because of this we always underestimate gross additions and cancellations in the queue. In general, this is to the disadvantage of the agent. Additions are always placed behind the agent's order, therefore underestimates pose no advantage. Cancellations, however, could happen in front of the agent's position, therefore underestimates give him a disadvantage.

4 Almgren and Chriss (2000)

I Explanation

We first look towards the model developed by Almgren and Chriss (2000)[2] as a starting baseline. Almgren and Chriss first define a trading strategy as a trade list, where given X units of a security, a time T where the the units must be liquidated, and N time intervals to trade in, the trading strategy will specify the number of units to sell at each time interval. In order to solve for the optimal trading strategy, Almgren and Chriss define the security price as a discrete arithmetic random walk given by

$$S_k = S_{k-1} + \sigma \tau^{\frac{1}{2}} \xi_k - \tau g(\frac{n_k}{\tau}) \tag{5}$$

where: S_k = price at time k

σ = volatility of the security

$\tau = \frac{T}{N}$ = length of discrete time interval

ξ_k = draws from independent normal random variables

$g(x)$ = permanent price impact function

$\frac{n_k}{\tau}$ = average rate of trading

Almgren and Chriss then model two different impacts for the price process. They introduce a temporary price impact function, $h(v)$. The intuition for this is that the permanent price impact is the effect from the trade on the supply and demand equilibrium of the security, while the temporary price impact is the adverse effect from absorbing the liquidity from the limit order book given a big enough trade volume. The temporary price impact does not affect the market price in the next period, but instead affects the price of subsequent trades for the order.

$$\widetilde{S_k} = S_{k-1} - h(\frac{n_k}{\tau}) \tag{6}$$

Combining the price process and impact functions, Almgren and Chriss define the expected shortfall and variance as:

$$E(x) = \sum_{k=1}^{N} \tau x_k g(\frac{n_k}{\tau}) + \sum_{k=1}^{N} n_k h(\frac{n_k}{\tau}) \tag{7}$$

$$Var(x) = \sigma^2 \sum_{k=1}^{N} \tau x_k^2 \tag{8}$$

For the model, Almgren and Chriss assume linears impact function, where

$$g(x) = \gamma x \tag{9}$$

$$h(x) = \epsilon sgn(x) + \eta x \tag{10}$$

The objective function that Almgren and Chriss use to solve for the optimal execution strategy is given below, where they aim to minimize the expected shortfall and variance of their orders given a risk-aversion parameter λ.

$$\min_x (E(x) + \lambda Var(x)) \tag{11}$$

The reason they minimize both expected shortfall and variance instead of just expected shortfall is because of the nature of the efficient frontier for optimal orders in their model, depicted in Figure 5.

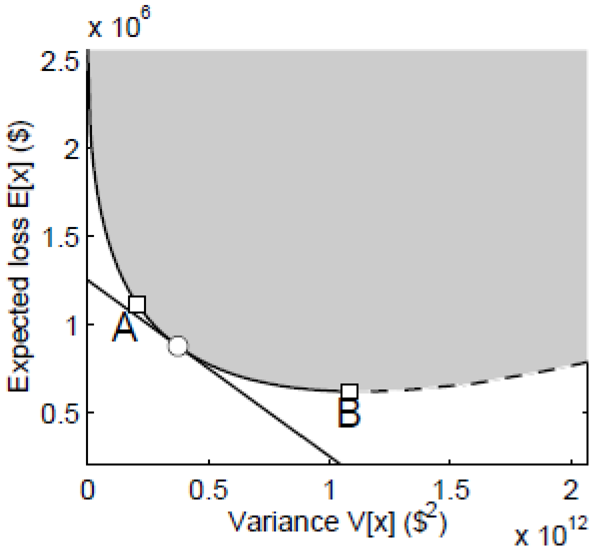


Figure 5: Almgren-Chriss efficient frontier, taken from Almgren-Chriss (2000)

The shaded region shows the attainable losses, while the solid curve shows the efficient frontier, where the expected shortfall is at a minimum given a variance level.

From this plot, Almgren and Chriss point out that simply minimizing the expected shortfall is not an effective approach, as shown in point B. Due to the nature of the curve, Almgren and Chriss argue that it is possible to reduce the variance by a large factor by incurring a slightly higher expected shortfall. Point A is a strategy with a high risk aversion.

Solving for the optimal execution strategies from Almgren and Chriss gets:

$$x_j = \frac{\sinh(\kappa(T - t_j))}{\sinh(\kappa T)} X,\ j = 0, ..., N \tag{12}$$

with the trade list given by

$$n_j = \frac{2\sinh(\frac{1}{2}\kappa\tau)}{\sinh(\kappa T)} \cosh(\kappa(T - t_{j-\frac{1}{2}}))X,\ j = 1, ..., N \tag{13}$$

where $t_{j-\frac{1}{2}} = (j - \frac{1}{2})\tau$

The expectation and variance of the optimal order execution strategy is

$$E(X) = \frac{1}{2}\gamma X^2 + \epsilon X + \widetilde{\eta} X^2 \frac{\tanh(\frac{1}{2}\kappa\tau)(\tau \sinh(2\kappa T) + 2T\sinh(\kappa\tau))}{2\tau^2 \sinh^2(\kappa T)} \tag{14}$$

$$V(X) = \frac{1}{2}\sigma^2 X^2 \frac{\tau \sinh(\kappa T)\cosh(\kappa(T-\tau)) - T\sinh(\kappa\tau)}{\sinh^2(\kappa T)\sinh(\kappa\tau)} \tag{15}$$

Given the optimal order strategies, we can see that varying different levels of risk aversion results in different trading behaviors. From Figures 6 and 7, we can see that a higher risk aversion parameter results in more aggressive early trades, while a lower risk aversion parameter results in slower beginning trade volumes.

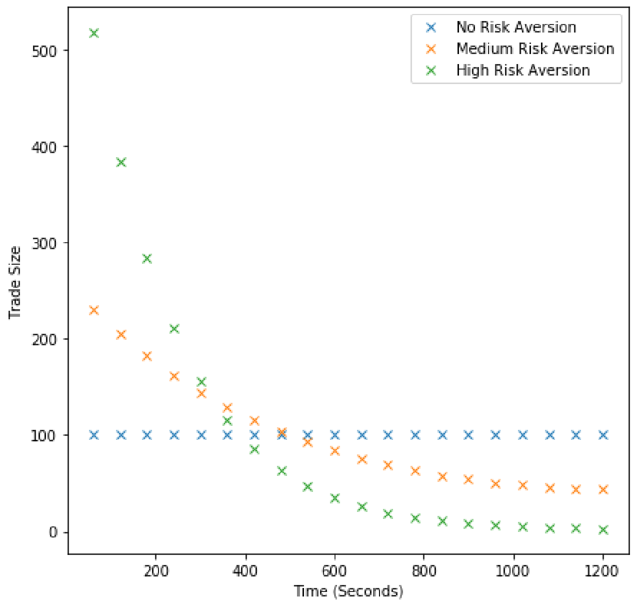


Figure 6: Almgren-Chriss Trade Sizes

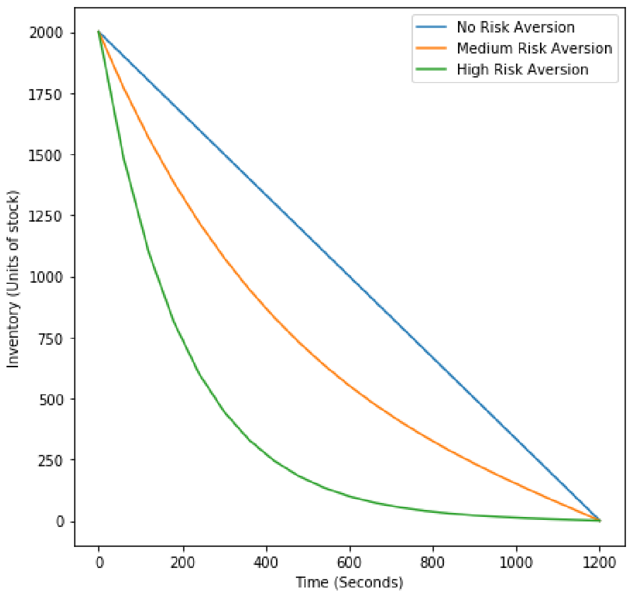


Figure 7: Almgren-Chriss Inventory Size

An interesting point about the Almgren and Chriss model is that the minimum expected shortfall strategy is to sell equal amounts of the security during the whole duration.

II Implementation Results

We implement Almgren and Chriss' (2000) model using our limit order book simulator. To test their model, we liquidate an inventory size of 2000 using a time interval between trades of 1 minute, which equates to a total trading duration of 20 minutes. We chose these parameters because the majority of historical trades are of size 100, so with these settings, our minimum expected shortfall strategy for Almgren and Chriss would be conducting sales of size 100.

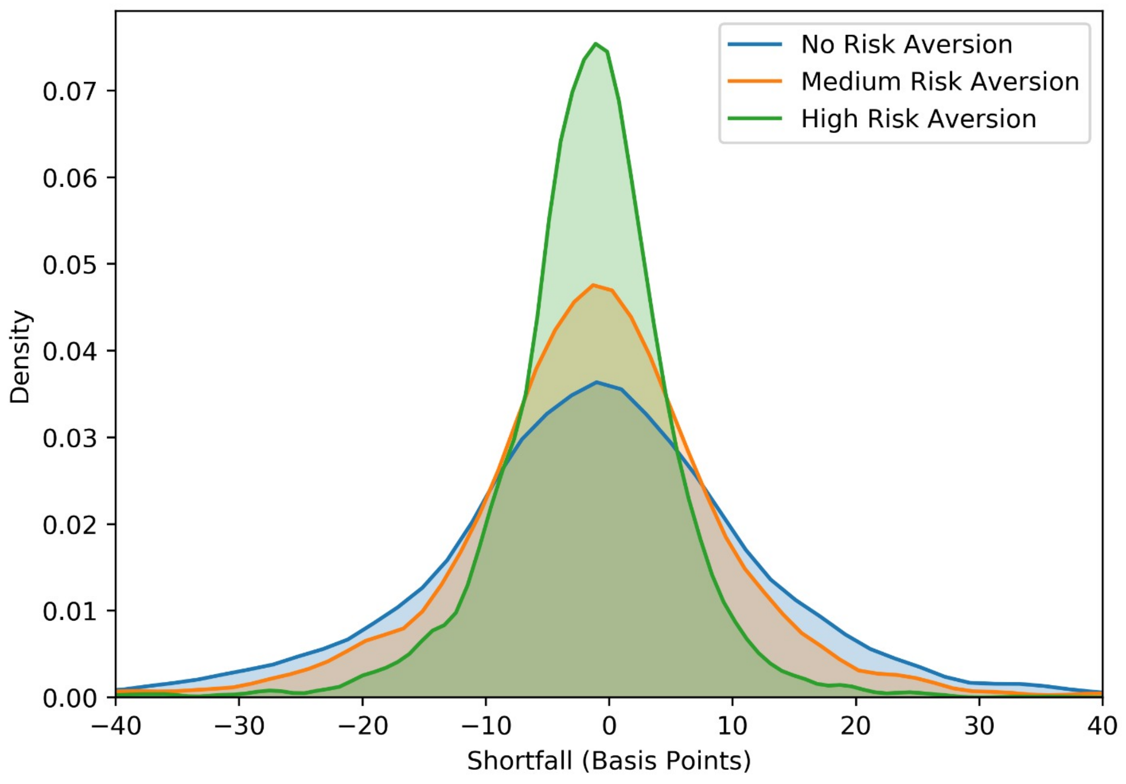


Figure 8: Density of implementation shortfall distributions following an Almgren-Chriss style execution schedule. Different distributions correspond to different risk parameters.

From the results presented in Figure 8 we can see that the variance of the shortfall decreases as risk aversion goes up. Table 1 below shows the expected shortfalls as well as its corresponding standard deviations for the different risk aversions.

Risk Aversion	Exec Speed	E(X) (bps)	SE(X) (bps)	Std(X) (bps)
None	Slow	-1.2987	0.2359	14.89
Medium	Medium	-1.3752	0.1800	11.36
High	Fast	-1.6221	0.1130	7.13

Table 1: Comparing shortfall and execution speeds in the Almgren-Chriss framework for different degrees of risk aversion.

5 Guo, Larrard, and Ruan (2017)

I Framework

In order to explore and implement a more recent and optimal-placement focused framework, we look towards Guo, Larrard, and Ruan (2017) for inspiration. Optimal order placement, the subject of the paper, is closely related to optimal execution in the sense that optimal execution is about splitting large orders in order to minimize price impact, while optimal placement deals with the timing for executing the smaller orders within a small time frame. Because of how closely the two are related, we cannot ignore this portion of the trading problem.

Guo, Larrard, and Ruan's (2017) propose a correlated random walk model for the bid/ask price of an asset. The model comes with its own set of assumptions, which are as follows -

- There is always a spread of 1 tick between the best bid and ask prices
- The best bid/ask price moves by 1 tick between successive time intervals
- The execution probability of limit orders is fixed in a trading period

.

The price model is modeled as a Markov chain as follows:

$$A_t = \sum_{i=1}^{t} X_i,\ A_0 = 0 \tag{16}$$

where A_t is the best ask price at time t and $X_t (1 \leq t \leq T)$ is a Markov chain with $\mathbb{P}(X_1 = 1) = \widetilde{p} = 1 - \mathbb{P}(X_1 = -1)$ and

$$\mathbb{P}(X_{i+1} = 1 | X_i = 1) = \mathbb{P}(X_{i+1} = -1 | X_i = -1) = p < \frac{1}{2},\ fori = 1, ..., T - 1 \tag{17}$$

This is considered to be the essential property of a correlated random walk. As the authors have stated in the paper as well, this is a simplistic case of the correlated random as described in Renshaw and Henderson [12] and in Gillis [13].

A key aspect to be noted here, and something which is observed in most empirical scenarios in this domain, is the choice of p. In order to exhibit the property of "mean-reversion", the value of $p < 0.5$. This makes a upward price move in the previous period more likely to be followed by a downward move in the current period, and vice-versa.

Another hyperparameter of the model is the execution probability of limit orders. As obvious, this execution probability, q, is a function of depth of the LOB, position of the order etc. However for our use-case, and as defined in the paper as well we can define a few rules for when q is valid (considering a limit order placed at price of k):

- if $A_t \leq k$ for some $t \leq T$, the value of $q = 1$
- if $A_t > k + 1$ for all $t \leq T$, the value of $q = 0$
- if A_t is k+1 and A_{t+1} is $k + 2$ then the limit order has some probability q for being executed between t and $t + 1$.

II Multi-Period Case

We now jump to the multi-period framework as described in the paper. In the most simple case, with which the section has been described, an investor needs to buy 1 share of stock by time T. Trades are allowed at and discrete time $t \in [0, T]$, and at time $t = T$, we execute the remainder of the order with a market order and the unexecuted limit order(s) are cancelled.

For the multi-period case, as the authors note, (A_t, X_t) have a Markov structure, resulting in the problem being a Markov Decision Process(MDP)[14]. Due to the essential property of an MDP, we can arrive at a solution to the problem recursively. We also know that each time t, the set of actions range from $A_t = \{Act^M, Act^L, Act^N\}$. In the context of placing bid orders, we now define a value function $V_t((A_t, X_t), \alpha_t)$, which is the cost of purchasing one share of stock (which was the investor's objective to begin with) by time T, taking action A_t.

As the paper mentions the total time frame of execution(0 to T) can be broken down into three parts via:

1. if $t < t_1^*, \alpha_t^*(-1) = Act^L,\ \alpha_t^*(1) = Act^N$;
2. if $t_1^* \leq t \leq t_2^*, \alpha_t^*(-1) = \alpha_t^*(1) = Act^L$;
3. if $t_2^* \leq t, \alpha_t^*(-1) = Act^M, \alpha_t^*(1) = Act^L$;

As evident here, there is a clear rule based system with respect to the action to be taken at time t based on the market movements from the previous to current state (1 being for an increase in the best ask price and -1 being for a decrease in the same). The formulae for t_1^* and t_2^*, as given in the paper, and as required very critically in our implementation as well, are as follows :

$$t_1^* = T - \left[\frac{1}{log(p-pq)} \cdot log\frac{q(1-2p)}{((1-p+pq)(f+2+r)-1)(1-q)(p^2q+1+p^2-2p)}\right] - 1 \tag{18}$$

and

$$t_2^* = T - \left[\frac{1}{log(p-pq)} \cdot log\frac{(f+r+1)(1-p+pq)-(1-p)(1-q)}{(1-p)(1-q)((f+2+r)(1-p+pq)-1)}\right] \tag{19}$$

Figure 9, taken from the paper, aims to provide a graphical explanation of the splitting of the entire time window into the requisite intervals, as mentioned above.

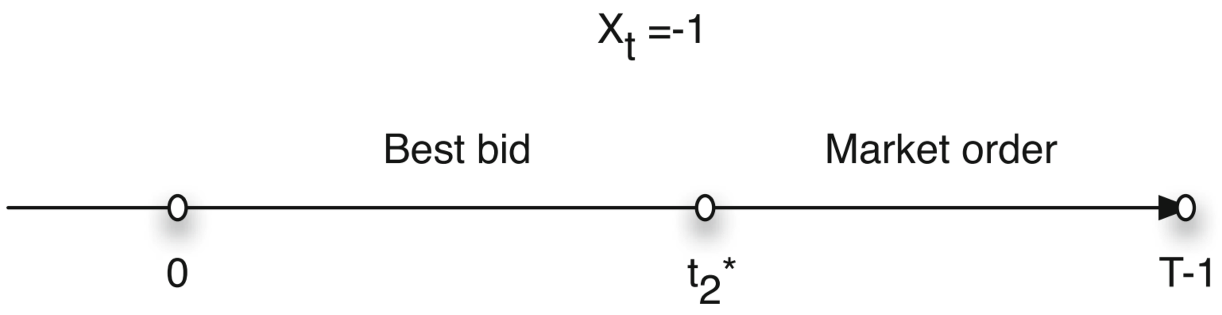


Figure 9: Optimal placement strategy for $1 < t \leq T-1$ if $X_t = -1$ [5]

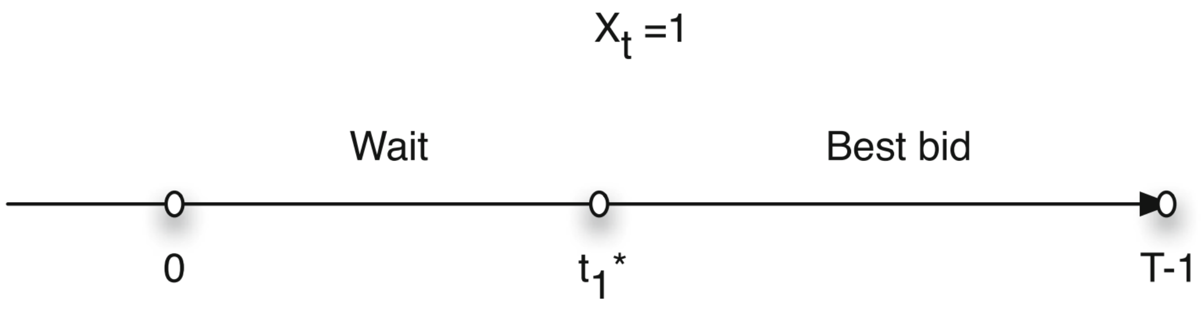


Figure 10: Optimal placement strategy for $1 < t \leq T - 1 if X_t = 1$ [5]

The last sub-section we talk about before moving on to the implementation, is the incorporation of price-impact in the the framework that has been established. In the paper, the authors state the problem by first remodeling the best ask price equation as : $A_{t+1} = A_t + c_1 m_t + c_2 I_t l_t + \in_{t+1}$ Here m_t and l_t are the sizes of the market and limit orders placed at time t. For for the purpose of simplicity, and for better modeling, c_1 and c_2 (the price impact due to market and limit order) are taken to be equal to one another, c. I_t is the flag of the limit order being executed/non-executed at time t. The final value function, for every time step t, is:

$$\begin{aligned} V(x, A_t) = min_{m_t, l_t \in [0,x]} \quad & m_t \left(f + c(m_t + l_t) + A_t\right) \\ & + p_l \left[l_t(-1 - r + c(m_t + l_t) + A_t) + (x_{mt} - l_t)(f + xc + A_t)\right] \\ & + (1 - p_l)(x - m_t)(f + xc + A_t) \end{aligned} \tag{20}$$

III Implementation

We now discuss the details of implementing the framework described in the paper - specifically the multi-period model, having price impact incorporated, with the best ask prices modeled as a correlated random walk.

With an initially specified maximum time duration of the trade, T, and a sampling frequency, S, as dictated by the order book, the implemented framework aims to optimize the equation written above for m_t or l_t at every time t. With T specified, the total duration is split up into N intervals ($N = \frac{T}{S}$).

For the values of the correlated random walk probability, p, and the probability of limit order execution, q, we can either find them empirically by fitting them for the previous execution cycle or provide an arbitrary that makes sense from the point of view of empirical observations. While most of the results were bases on the latter approach, the former was tried as well.

In order to keep the broad figures similar through the course of our analysis, we aim to buy 2000 shares of Apple stock in a time duration initially specified to be 20 minutes.

The first result we observe is for implementing this framework on 4000 randomly sampled Apple stock price paths.

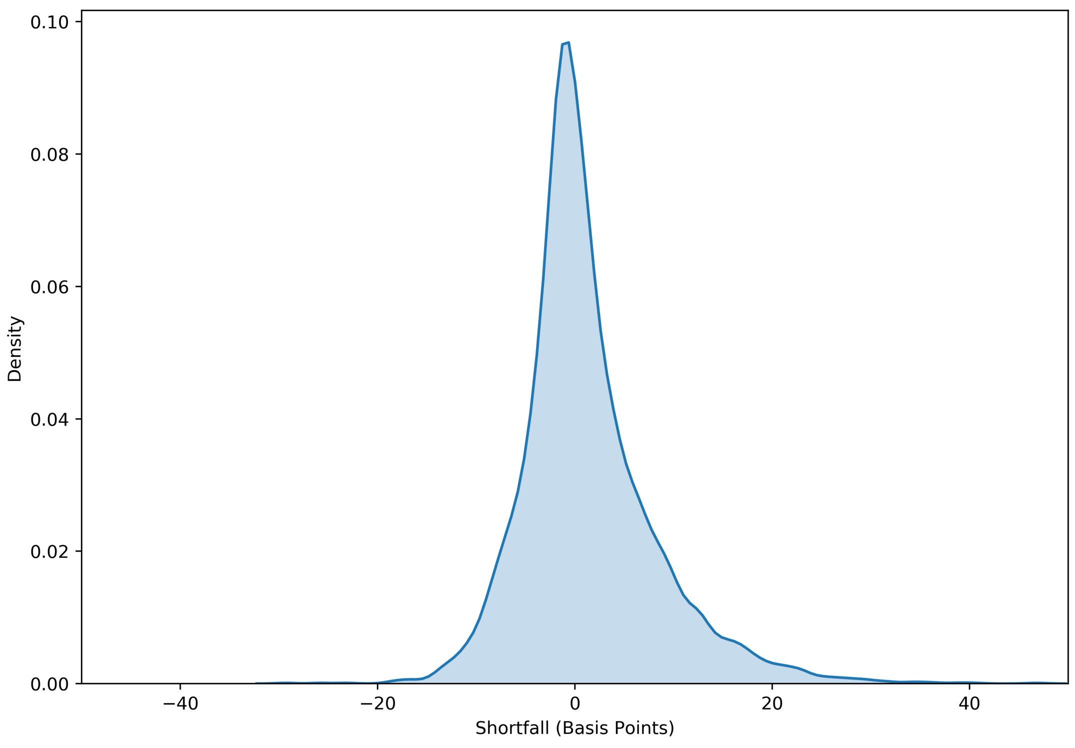


Figure 11: Shortfall(in basis points) density plot for 4000 trials

E(X) (bps)	Std(X) (bps)	Skew	Kurtosis (Exc.)
1.2933	6.7372	1.136	3.331

Table 2: Moments of the distribution of shortfalls in Fig. 11

The slightly positive expected shortfall can be considered to be a corollary to the approximations made to arrive at a slightly abridged form of the model, compared to the exact framework described in the paper. For one, the model does not incorporate any market order fees or limit order rebates, something which actually forms a consistent part of the original framework. This itself makes the model less inhibiting in terms of crossing the spread and placing market orders repeatedly instead of placing limit orders.

However, with that being said, we do achieve a performance with the model which, over 4000 trials, is sufficient to claim a much superior performance to the baseline Almgren-Chriss Framework.

Another interesting visualisation could be on the analysis of the expected shortfall and the shortfall distributions with respect to the maximum specified trade duration (T). Previous analyses dealt with T being set to 20*60s(20 minutes). Now since the time window forms an important part of the optimal strategy, as we could see above (dividing the time T into t_1^* and t_2^*), we now look at simulating the framework for a range of T values.

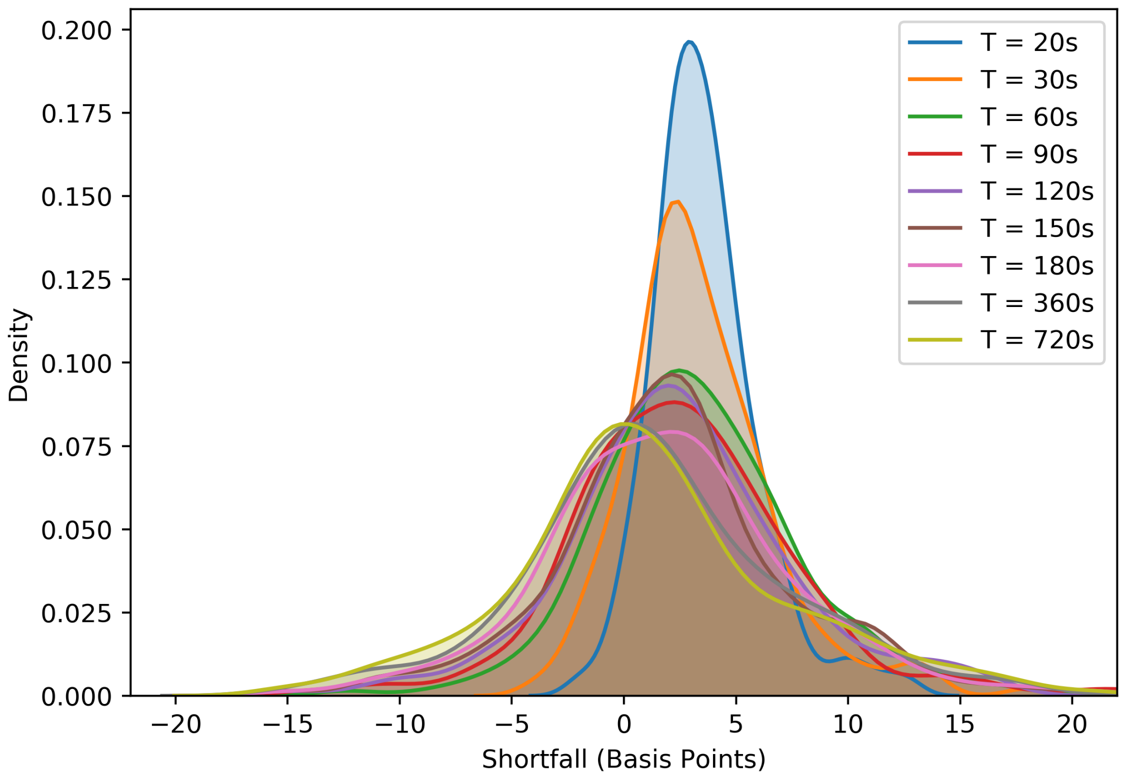


Figure 12: Shortfall(in basis points) distributions for different values of trade duration

The table below summarises the expected shortfalls and variances.

T Value	20s	30s	60s	90s	120s	150s	180s	360s	720s
E(x) bps	3.628	3.536	3.318	2.594	2.461	2.293	1.933	1.539	1.241
Std(X) bps	2.391	3.748	4.612	5.021	5.049	5.580	5.396	5.976	6.169

Table 3: Expected shortfalls and variances of distributions from Fig. 12

Our qualitative and quantitative observations, from Fig.12 and Tab.3, confirm our inference from Fig.11. We are able to see higher(on the positive side) shortfalls with lesser duration times as the framework, in uninhibited way, places large market orders to fulfill the total order when close to T. This would be not be the case in the presence of a fee structure for market orders and a rebate structure for limit orders, as then the optimal solution to the value function would involve smaller market orders and/or higher cost of trade due to the fees. The trials with higher trade duration show more expected behaviour due to a mix of limit and market orders placed at different time intervals to fulfill the entire initial order.

6 Reinforcement Learning

I Motivation

In reinforcement learning, an agent observes its environment, makes an action, then receives feedback from its environment in the form of a reward based on the result of its action, as well as a new state in the environment. In essence, it is a Markov Decision Process. Throughout the years, many academics have developed Markov models for the limit order book, such as Abergel and Jedidi (2015)[7], Kelly and Yudovina (2017)[15], as well as Guo, Larrard, and Ruan (2017)[5] which we implement in this paper. As such, it makes sense to extend the optimal order execution problem using reinforcement learning.

Additionally, it is quite natural to cast a trader and the market into the classic reinforcement learning framework of an agent and an environment. By choosing an adequate definition of reward and action, and using a realistic simulator for the market, it ought to be possible to create an agent which is able to trade optimally and dynamically with the market.

II Framework

II.1 Outline & Scope

We use the limit order book simulator described in Section II of the Methodology as the simulated *environment* for the agent to interact with. The agent observes the environment as a set of values including microstructure data, prices, time progression and his own inventory. These values are referred to as the *state* of the environment. In the environment the agent has the ability to place limit orders, market orders or simply doing nothing. The agent's action is evaluated against a *reward* function. In the learning process, the agent uses the reward function in combination with a learning algorithm to derive a policy, which is a mapping from state to action space. The reinforcement learning framework outlined in this paragraph is summarized in Figure 13.

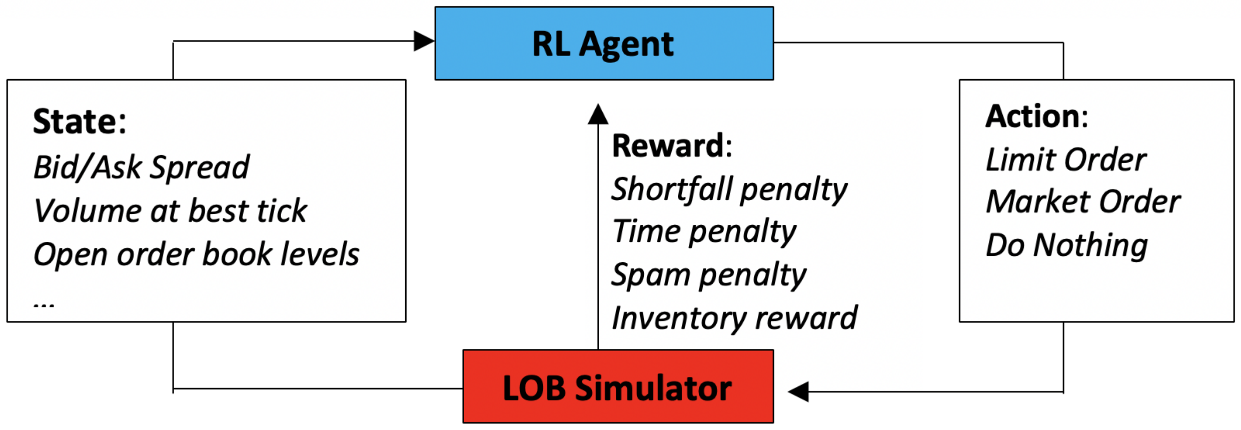


Figure 13: Diagram of the reinforcement learning framework. A feeback loop is facilitated through propagation of *state*, *action* and *reward*.

Based on different reward functions, an agent can end up with very different policies and behaviours. We explore this by varying parameter weights in the reward function to derive four different agents. We thus show how the optimal execution behaviour of an agent changes with different definitions of risks and rewards.

II.2 Problem and Objective

Our agent needs to liquidate 2000 AAPL shares within 20 minutes. The agent will observe its environment every five seconds, and decide on an action to take. The intuition behind giving our agent this amount of freedom is so that it can find the optimal times to place orders instead of being forced to place periodic orders as in Almgren and Chriss. The objective will be to minimize the incurred shortfall from its trades. If the agent is not able to liquidate all of its assets during its allotted time, it will be forced to place a market order on its remaining inventory. Ideally, the agent will be incentivized to try to complete its task on time, and with a minimal amount of shortfall.

II.3 Action Space

The agent is limited to only trading one side of the market at a time. It cannot buy and sell at the same time. In our case, the agent will only be operating on the sell side of a trade. During each instance, the agent will observe its state and decide to either place a limit order of size 100 on the second level of the limit order book, place a market order of size 100, or do nothing. By limiting the actions of our agent, we hope to have the agent focus more on the timing of its order placements instead of the execution.

II.4 State Space

The state space of our agent is as follows:

1. Volume at best tick.
2. Volume at second best tick.
3. Percentage of trading day elapsed.
4. Percentage of the 20-minute trade window elapsed.
5. Change in inventory since last observation.
6. Percentage different from the initial price of the epoch.
7. Gross trade volume since the last observation.
8. Net trade volume since the last trade observation.
9. Bid-Ask Spread.
10. Number of open orderbook levels.
11. Whether the agent has a limit order at the second tick.

II.5 Reward

The general reward function R_t for our agent is as follows:

$$\text{shortfall-penalty}_t = V_t \cdot \frac{P_t - P_0}{P_0} \tag{21}$$

$$\text{time-penalty}_t = -\,\mu \cdot S\Big(\frac{inventory_t}{V_{MO}} - \frac{T-t}{\Delta T}\Big) \tag{22}$$

$$\text{spam-penalty}_t = -\,\gamma \cdot I\Big(\sum_{n=0}^{N} V_{t-n\Delta T} > \phi\Big) \tag{23}$$

$$\text{inventory-reward}_t = ,\nu \cdot I(V_t > 0) \tag{24}$$

$$R_t = \text{shortfall}_t + \text{time-penalty}_t + \text{spam-penalty}_t + \text{inventory-reward}_t \tag{25}$$

Where P_t is the best bid at time t, V_t is the order volume executed at time t, T is the duration of the trading window, ΔT is the duration of one time step, V_{MO} is the volume of market orders executed by the agent, μ, γ, ν are scaling factors, ϕ is a threshold parameter, $I(...)$ is the indicator function, $S(...)$ is the sigmoid function.

While at first inspection the reward function may appear arbitrary and overly complex, each penalty and reward term aims to promote a different aspect of positive trading behavior.

The *shortfall-penalty* is the most straightforward term, it just calculates the volume weighted execution shortfall of an order. The less desirable the execution price is, the larger is the penalty.

The *time-penalty* is something we found very useful to achieve faster convergence when training the reinforcement learning agent. If the agent hasn't completed his trading behaviour by the end of the trading window, the remaining inventory is executed in full against the order book, this is facilitated through the shortfall-penalty. Since this penalty is very abrupt and weighted towards the end of the trading window, the time-penalty aims to promote greater foresight on part of the agent. The time-penalty starts awarding penalties several periods ahead of the final period. It does so when the remaining inventory cannot be fully executed aggressively through market orders within the remainder of the trading window. For smoothing purposes, the sigmoid function is used. The penalty should therefore encourage the agent to aggressively decrease inventory when approaching the final trading periods.

The *spam-penalty* is a cost which gets triggered when the agent executes market orders at too high rate. By spacing out orders the agent should reduce the risk of adverse market impact. While this behaviour should be promoted by the shortfall-penalty, it is helpful for the training process to add a term specifically for this. Agents with large spam-penalty terms are expected to have a stronger reliance on limit order execution and end up utilizing larger amounts of the trading window, also giving the limit orders more opportunity time to get hit.

The *inventory-reward* is a positive reward for reducing inventory, usually through market orders. As long as a trader holds assets which he aims to get rid of, he carries inventory risk. This is the risk of an adverse price movement while he is holding the assets. The agent should be getting a reward for reducing inventory risk. By attenuating ν, we can tune the agent's affinity for reducing inventory, even at the prospect of accepting larger execution shortfall. The inventory-reward term is therefore closely related to promoting behaviour which is described as "risk-averse" in the Almgren-Chriss framework.

By experimenting with the reward scaling parameters μ, γ, ν, we are able to create training agents with very different execution behaviors. For example, by turning down the spam-penalty and increasing inventory-reward, we would expect to create an agent who tries to execute his inventory at the beginning of the trading window using market orders. On the other hand, an agent with increased spam-penalty and small inventory reward should mainly be aiming to execute with limit orders, leading to a larger variance in shortfall, but with the prospect of executing orders at favorable prices.

III Implementation

Our implementation of the reinforcement learning process relies mainly on the Proximal Policy Optimization algorithm (PPO2) (Schulman et al. 2017)[16], using the wrapper implemented by OpenAI.

PPO2 builds on the class of Trust Region Policy Optimization (TPPO) algorithms (Schulman et al. 2017)[16] which use a surrogate loss objective function in the form of a likelihood ratio, constrained by the size of the policy update. In general, these methods perform one gradient ascent update per data sample, whereas PPO allows multiple gradient updates through minibatch sampling.

We elect to use a multilayer perceptron (MLP) as our choice of policy. Although a long-short term memory (LSTM) may be more appropriate given the domain analogue to a time series problem, the MLP is much less computationally intense which allows us to refine the agent over a greater number of epochs.

IV Results

We train four agents on data samples from 2012-2014 with different reward scaling factors in order to see the effects of prioritizing different components of our reward function. The scaling factors for each agent are shown in Table 4. Note that the time-penalty was kept constant for all agents as it generally helped with convergence.

Agent	γ (*Spam*)	μ (*Time*)	ν (*Inventory*)
A	0.0002	0.0004	0.0002
B	0.0000	0.0004	0.0002
C	0.0002	0.0004	0.0000
D	0.0000	0.0004	0.0000

Table 4: Agent Reward Function Parameters

IV.1 Shortfall Distributions

By applying our agent to the hold-out data set of 3000 trading windows from 2015, we are able to obtain a distribution of shortfalls for each agent's trading performance in the simulator. The shortfall distributions are shown in Figure 14. The moments of each agent's shortfall distribution are summarized in Table 5.

Agent B and D achieved the lowest variance in shortfalls. Their execution performance is almost identical to one another. Agent C achieved the largest variance. While the mode of the shortfall distributions increase with decreasing variance, the expected shortfall does not improve with increasing variance. The reverse appears to be true, Agent C has the largest variance and also the largest expected shortfall. This may be attributed to the fact that it has the largest skew and excess kurtosis of all the distributions. While Agent C may achieve the best execution price more often than the other Agents, on average, it will perform worse.

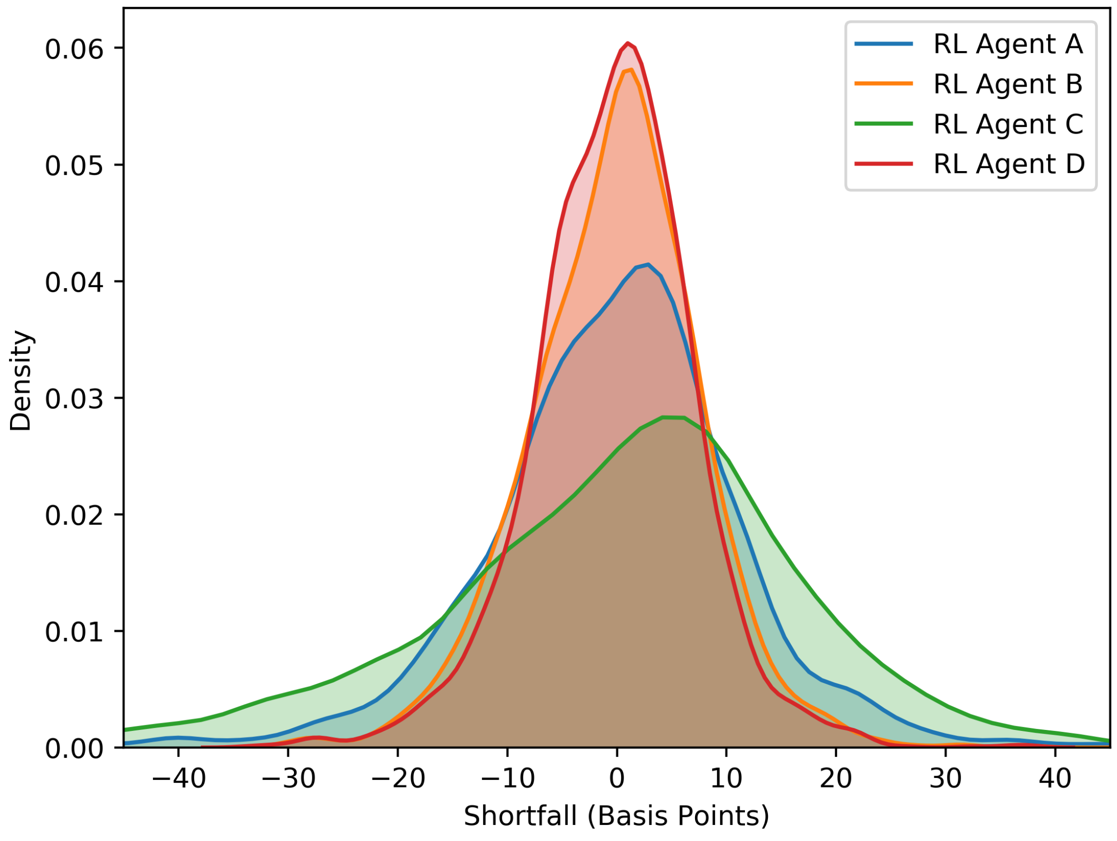


Figure 14: Density plots of the shortfall distribution for reinforcement agents A, B, C and D. Evaluated from 3000 hold-out samples. Agent B's distribution overlays closely with that of Agent D.

Agent	E(X) (bps)	Std(X) (bps)	Skew	Kurtosis (Exc.)
A	-0.2450	12.1996	-0.3522	3.6306
B	-0.1097	7.9113	-0.0206	1.4787
C	-0.6137	18.4442	-1.2747	5.8946
D	-0.2879	7.3768	-0.1264	1.7005

Table 5: Moments of the distribution of shortfalls achieved by the four reinforcement learning agents on the 3000-sample hold-out data set. A density visualization of the distributions can be found in Figure 14

IV.2 Trading Behaviour

We then analyze the performance dynamics of the agents by investigating their action choices, inventory size and trading duration over time. Explicitly, for the 3000 hold-out samples we have averaged these metrics over time and visualized them in Figures 15 through 18. The top plots show the proportion at which an action (*Place Limit Order*, *Place Market Order*, *Do Nothing*) was chosen. Bottom shows the inventory size and the fraction of agents still trading.

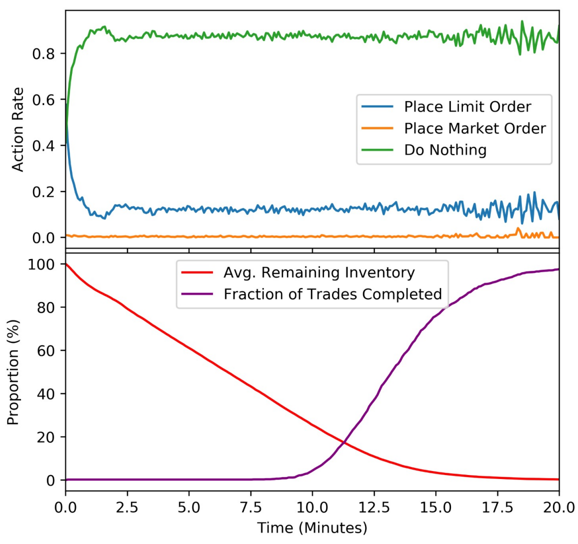


Figure 15: Agent A (Balanced Parameters)

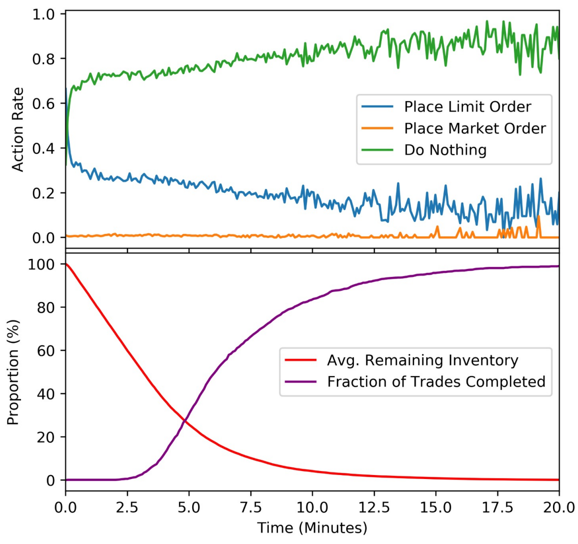


Figure 16: Agent B (Spamming Allowed)

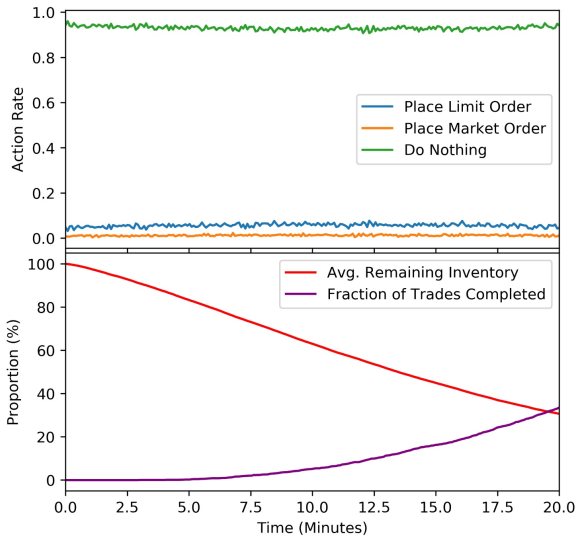


Figure 17: Agent C (No Trade Reward)

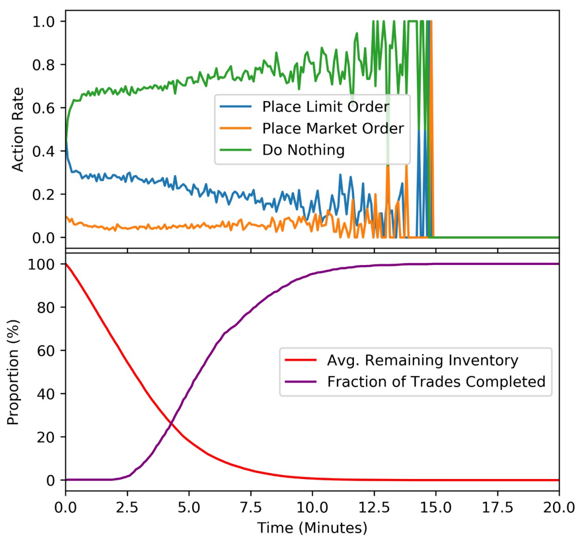


Figure 18: Agent D (No Reward + Spam)

From comparing Agent A and B, we can see that relaxing the order spamming constraint resulted in a more aggressive agent with larger propensity for placing trades. As a result Agent B liquidates inventory much faster than Agent A. For agent B the first agents start completing their order schedule at the 4 minute mark, for Agent A this only starts occurring at 11 minutes.

Agent A, B, and C agree on a strategy of placing limit orders: they place limit orders at higher rate at the start of the trading session than later in the session. This makes sense as they are creating an initial stacking of the limit order book. After this initial stack of trades is placed, it isn't favorable to keep spamming limit orders at an already saturated order book. Agent C does not trade in this way, which may depend on the lacking reward for reducing inventory.

Agent B has a very similar liquidation speed as Agent D, the agent carrying mostly about shortfall. It makes sense that the shortfall distribution for these two agents are close to one another as we've seen in Figure 14. An interesting difference, however, is that Agent D completes virtually all trading sessions after 15 minutes (75% of the allocated time), while for other Agents it is more likely to use the full trading window.

Removing the inventory-reduction-reward in Agent C, we observe that the agent has less incentive to place trades and instead performs *Do Nothing* more often than any other agent. This results in an average of about 30% of trade inventory not being executed at the end of the trading window. The remaining trade inventory is executed as a market order against the limit order book, locking in execution at undesirable price levels. This helps explain why Agent C had the worst expected shortfall out of all the agents and motivates the inclusion of an inventory reduction reward to promote trading.

Removing both the spamming constraint as well as the inventory direction reward results in Agent D, which tries to liquidate as fast as possible. This is an interesting outcome since from our previous agents, we can assume that without the inventory reduction reward, the agent would have a much lower incentive to liquidate quickly.

7 Discussion

In this section, we compare our results from the three different approaches to the problem, including the advantages and disadvantages of each.

I Comparison of Results

Judging by the average shortfalls for each strategy, the Almgren and Chriss model results in the highest amount of shortfall. This makes sense, since the Almgren and Chriss model only executes market orders, which are not as profitable as limit orders. Judging by expected shortfall only, all RL agents beat Almgren-Chriss.

Xin Guo's model, however, sports a positive expected shortfall, which is a surprising result. This implies that Xin Guo can expect to sell assets above market value. There are some explanations for why we might achieve such a result. For one, we only implement a simplified version of the Guo model. The full model accounts for transaction costs of market orders as well as order rebates for limit orders. Additionally, we don't correct for a drift term in the stock price process. When testing the execution model over many samples and long time frames, a positive drift would work in favor of a selling agent. To test this hypothesis, we could implement the Guo model for a buy agent, in this case the drift would be adverse to the agent's desired price movement and so the expected shortfall would be negative.

Another benchmark we can select execution models by is if the combination of mean shortfall and shortfall variance is on the efficient frontier, as described in Almgren-Chriss (2000) and depicted in Figure 5. In Figure 19 we create our own version of the plot where we place the Guo, Almgren-Chriss and reinforcement learning results on the mean-variance grid. From this plot it is evident that out of all execution frameworks Guo is the superior model. It has the smallest variance as well as the best expected shortfall. Compared to Almgren-Chriss, at least the leftmost two of the reinforcement learning agents (the ones with the lowest variance) are superior in mean-variance space.

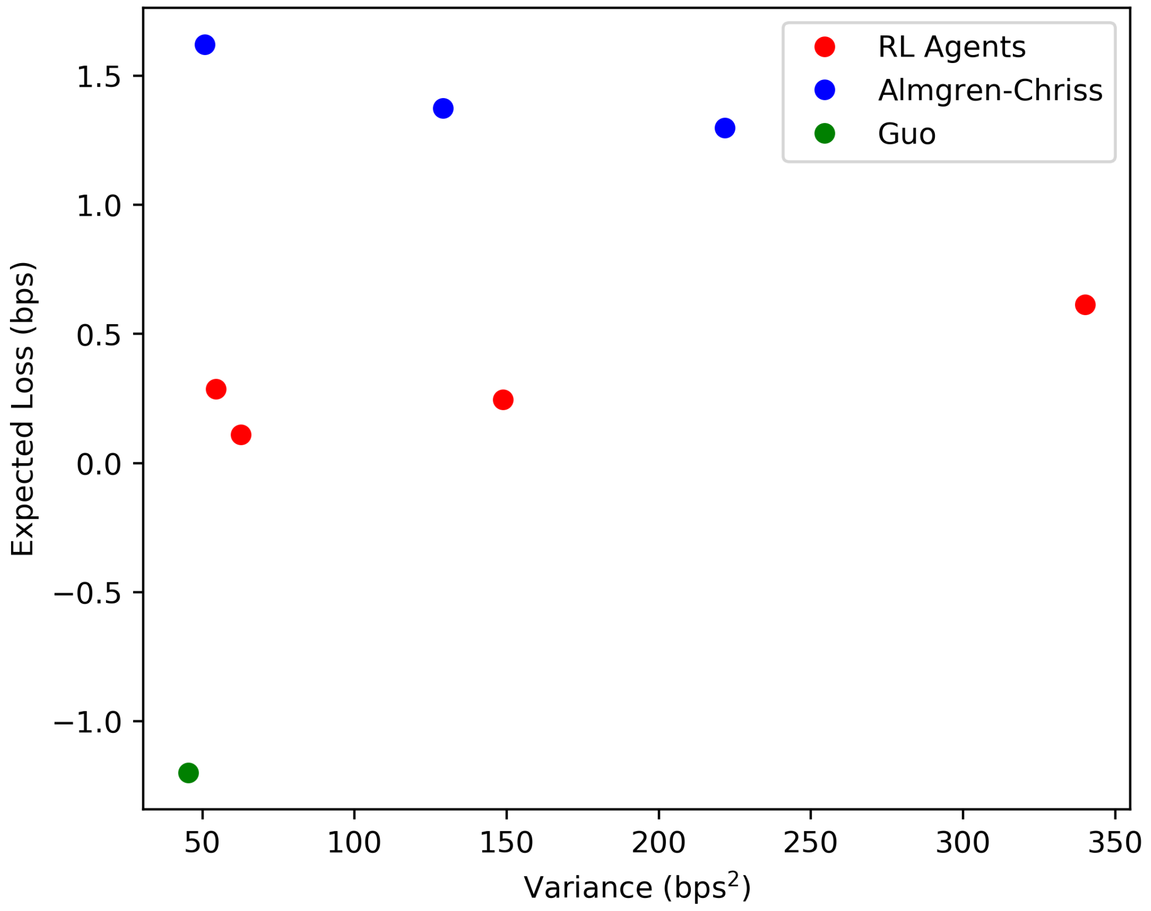


Figure 19: The location of execution results in mean-variance space. Note that for consistency with Figure 5, the y axis is loss, which is the negative of shortfall. Results which are to the bottom left corner are more desirable.

It would be of aid in contextual understanding, to visualise all the shortfall density distributions together.

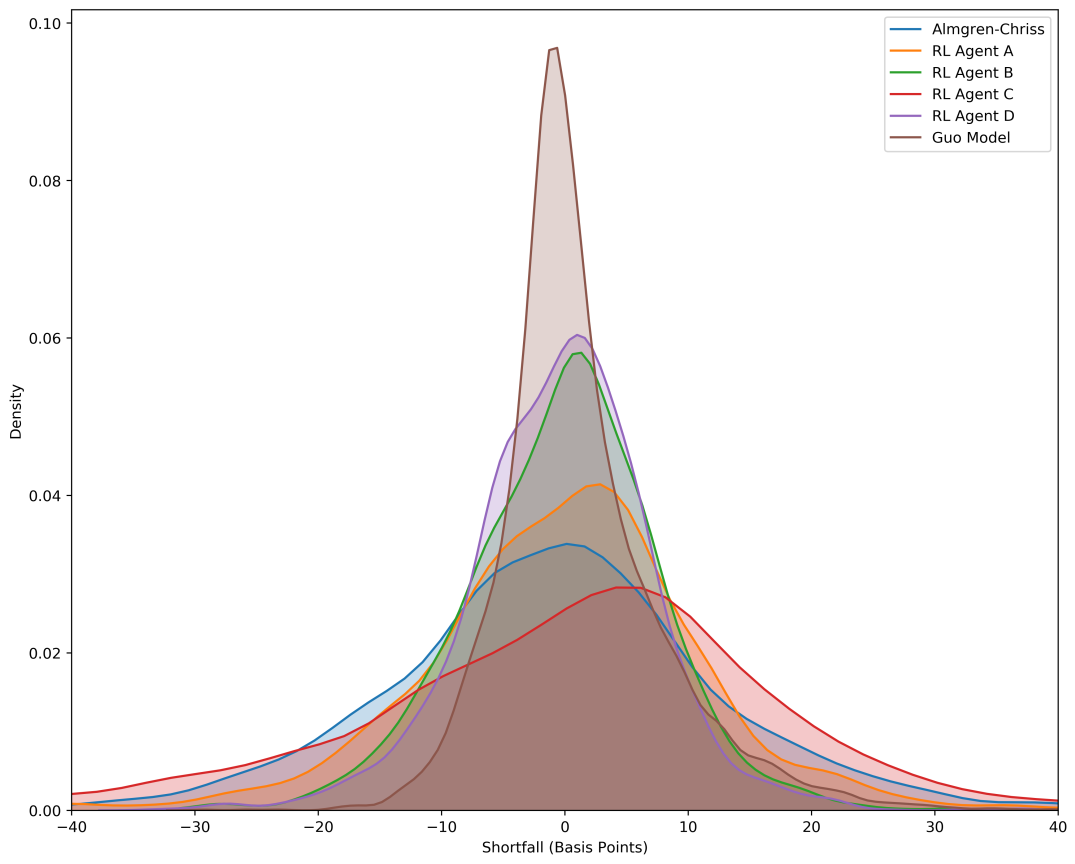


Figure 20: The shortfall from execution of the Algren-Chriss framework, the 4 Reinforcement Learning agents and the Guo Model

It is hence clearly evident that while RL agents outperform the Almgren-Chriss framework consistently, the Guo Model offers the best execution quality over the same number of trials. Needless to say, and as mentioned before, the Guo Model here is definitely also subject to a range of assumptions made while implementing the same.

II Reinforcement Learning Shortcomings

While we were able to create a functioning set of reinforcement learning agents that perform order execution strategies, this approach comes with significant downsides when looking to implement it in production.

For one, it is difficult to define a reward function which leads to behaviour a human would deem as sensible. Simply optimizing shortfall doesn't lead to stable behaviour. It took significant experimentation to stop agents from just executing continuously in the market. Looking at the reward function, it isn't clear how a practitioner would select or tune an agent with the risk/reward characteristics they deem optimal for their company.

Furthermore, even though we were somewhat able to illuminate the average trading behavior of an agent in Figures 15 through 18, the agent is still much closer to a black-box than a white-box. Additional explainability analysis needs to be performed to evaluate the robustness of the agent's behaviour in a trading environment. While tools such as Shapley values are useful for shining more light on the black-box issue, it doesn't get around the fact that the agent may act unpredictably once placed in an unobserved market regime. Since the underlying policy model of our agent is a neural network, it would surprising if the agent behaviour generalizes well outside of the domain of observed values.

Finally, while reinforcement learning performs very well in controlled environments like games, which are readily, if not already, virtualized, it is questionable how well it performs for financial applications where our simulators are only approximations to the real world.

The above points raise serious concerns for implementation issues of a live trading execution system based on reinforcement learning. Perhaps rather than using the agents to place live orders, it makers more sense to use them as an analysis tool to come up with sensible execution strategies in an offline setting. For example, one strategy which seemed to have payed off for the reinforcement learning agents is stacking of limit orders at the beginning of the execution window. This insight has valuable merit and doesn't need a reinforcement agent in a live market to execute.

8 Conclusion

In this paper, we have constructed a market simulation approach to train a reinforcement learning agent to optimally execute trading instructions, and we have compared its performance to standard models in the literature.

By varying the reward function we successfully implemented four reinforcement learning agents with unique trading characteristics. All the agents outperform the Almgren-Chriss benchmark while having unique trading styles. Although the agent performance relative to the Guo model is more ambiguous, the Guo model also relies on more simplifying assumptions than our market simulation, suggesting efficacy in our approach despite implementation drawbacks.

As discussed, further research could investigate improvements to the market simulation approach, or even use live data. Additionally, we have opted to remain relatively simple in our definitions of the reinforcement learning space. It may be possible to achieve even better results by using an expanded state space or a more delicately crafted reward function, and it would be pertinent to investigate the use of a larger action space before any reinforcement learning agent is implemented in practice.

References

[1] D. Bertsimas and A. W. Lo, "Optimal control of execution costs," *Journal of Financial Markets*, 1998.

[2] R. Almgren and N. Chriss, "Optimal execution of portfolio transactions," *Journal of Risk*, 2000.

[3] A. Obizhaeva and J. Wang, "Optimal trading strategy and supply/demand dynamics," *Working Paper*, 2005.

[4] C. Benazzoli and L. D. Persio, "Optimal execution strategy in liquidity framework," *International Journal of Applied Mathematics*, 2014.

[5] X. Guo, A. de Larrard, and Z. Ruan, "Optimal placement in a limit order book: An analytical approach," *Mathematics and Financial Economics*, 2017.

[6] B. Hollifield, R. A. Miller, and P. Sandas, "Empirical analysis of limit order markets," *The Review of Economic Studies*, 2004.

[7] F. Abergel and A. Jedidi, "Long-time behavior of a hawkes process–based limit order book," *SIAM Journal on Financial Mathematics*, 2015.

[8] W. Huang, C.-A. Lehalle, and M. Rosenbaum, "Simulating and analyzing order book data: The queue-reactive model," 2013.

[9] S. Vyetrenko, D. Byrd, N. Petosa, M. Mahfouz, D. Dervovic, M. Veloso, and T. H. Balch, "Get real: Realism metrics for robust limit order book market simulations," 2019.

[10] S. Vyetrenko and S. Xu, "Risk-sensitive compact decision trees for autonomous execu-

tion in presence of simulated market response," *ICML 2019 Workshop on AI in Finance*, 2019.

[11] T. Spooner, J. Fearnley, R. Savani, and A. Koukorinis, "Market making via reinforcement learning," 2018.

[12] E. Renshaw and R. Henderson, "The correlated random walk," *Journal of Applied Probability*, 1981.

[13] J. Gillis, "Correlated random walk," *Mathematical Proceedings of the Cambridge Philosophical Society*, 1981.

[14] O. Alagoz, H. Hsu, A. J. Schaefer, and M. S. Roberts, "Markov decision processes: A tool for sequential decision making under uncertainty," *Univ. Of Pittsburg*, 2010.

[15] F. Kelly and E. Yudovina, "A markov model of a limit order book: Thresholds, recurrence, and trading strategies," *Working paper*, 2017.

[16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.